DevOps Case Study Report

Project: simple-webapp-jenkins-ci-cd | Author: Punith C | Date: November 27, 2025

Executive Summary

This report presents a complete CI/CD pipeline for a static web application using Jenkins. The pipeline automates checkout, validation, testing, and Docker-based deployment. The final build (#10) achieved 100% (4/4 stages) success and deployed a containerized app accessible at http://localhost:8090.

Problem Statement & Objectives

Create a CI/CD pipeline for a static HTML/CSS/JS website using Jenkins and implement automated deployment. Objectives included ensuring reproducible builds, minimal manual steps, and clear pipeline visibility.

Objectives

- Automate Checkout → Build → Test → Deploy
- Validate required files before deployment
- Use Docker/Nginx for reproducible deployment
- Document results and lessons learned

Scope

- Static website: index.html, styles.css, script.js
- Jenkins declarative pipeline (4 stages + post)
- Dockerfile for Nginx-based container
- Optional docker-compose for local Jenkins + webapp

Environment & Architecture

Windows host with WSL2 Ubuntu as the Jenkins agent. Docker used for building and running the containerized web app. Architecture flow: Developer (GitHub) → Jenkins Pipeline → Docker Image → Nginx Container → Browser.

Technologies Used

- HTML5, CSS3, JavaScript
- Nginx (nginx:alpine)
- Docker
- Jenkins (Declarative Pipeline)
- Git/GitHub

Tools Used

- Jenkins 2.528.2
- Docker 28.2.2

- Git 2.43.0
- Windows + WSL2 (Ubuntu 24.04)
- VS Code

Methodology / Implementation Steps

- Create app files and Dockerfile; initialize Git repo
- Configure Jenkins (Pipeline from SCM; set script path)
- Make scripts executable; add Jenkins to docker group; restart Jenkins
- Run validate.sh in Build & Test stages
- Build Docker image and run container on port 8090
- Post-deploy verification and logging

Jenkins Pipeline

- Stages: Checkout, Build (validate), Test (validate), Deploy (docker)
- Environment: DEPLOY_METHOD=docker; CONTAINER_NAME=simple-webapp-demo; WEBAPP_PORT=8090
- Nested repo path handled with dir() blocks

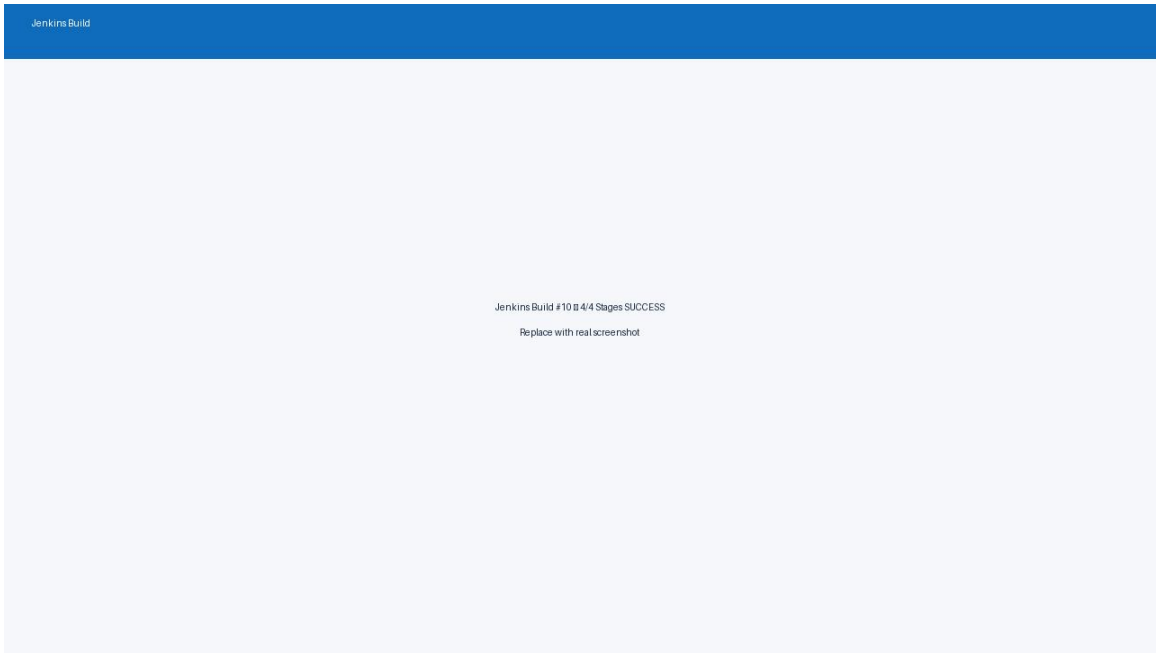
Deployment

- Image: simple-webapp:latest (Image ID: e54b76dbdf5f)
- Container: simple-webapp-demo (ID: 1db05a33b511)
- Port mapping: 8090 → 80; URL: http://localhost:8090
- Automated stop/remove old container before redeploy

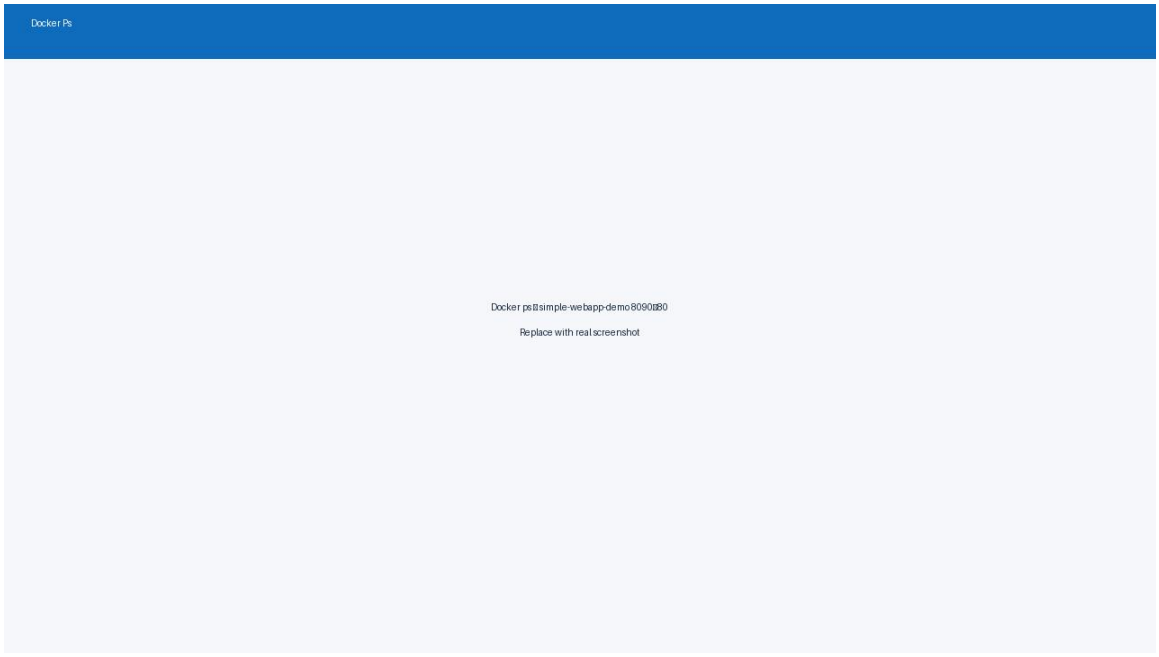
Results & Metrics

- Build #10: SUCCESS (4/4 stages)
- Total runtime ~15 seconds
- Application functional; JS alert verified

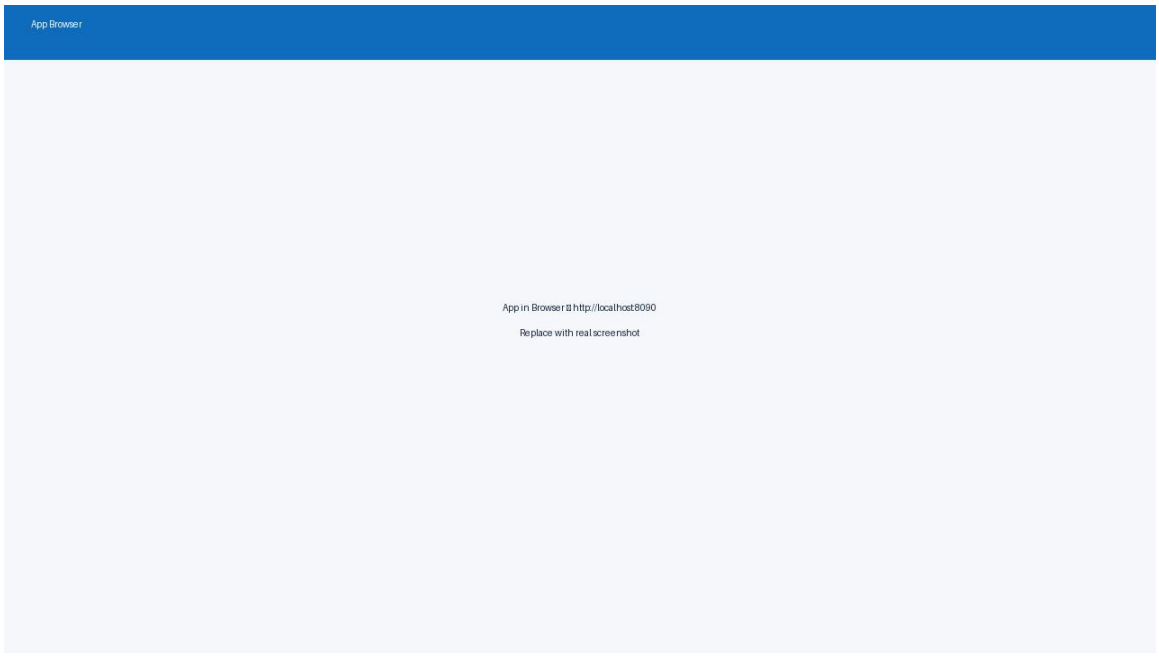
Screenshots



Jenkins Build #10 — All stages green



Docker container simple-webapp-demo (8090→80)



App at http://localhost:8090 with JS alert

Source Code Overview

- Jenkinsfile — Declarative pipeline (4 stages + post)
- Dockerfile — Nginx-based static site image
- validate.sh — Pre-deploy file checks
- deploy.sh — Copy-based deploy alternative
- index.html, styles.css, script.js — Static app

Selected Code Listings

Jenkinsfile

```
// Jenkinsfile — Declarative pipeline for Simple WebApp CI/CD demo
pipeline {
  agent any

  // Environment variables (override in Jenkins configuration or pipeline
parameters)
  environment {
    // Set REPO_URL in Jenkins credentials or pipeline environment if you want
the pipeline to clone externally
    REPO_URL = ""
    DEPLOY_METHOD = "docker" // options: 'copy' or 'docker'
    IMAGE_NAME = "simple-webapp:latest"
    CONTAINER_NAME = "simple-webapp-demo"
    WEBAPP_PORT = "8090" // External port for accessing the webapp
  }
```

```
  stages {
    stage('Checkout') {
      steps {
        script {
          if (env.REPO_URL?.trim()) {
            echo "Cloning from ${env.REPO_URL} (branch: main)..."
            sh 'git clone --branch main ${REPO_URL} repo || true'
            dir('repo') { sh 'ls -la' }
          } else {
            echo 'REPO_URL not set — using current workspace contents.'
            echo 'Changing to project directory: simple-webapp-jenkins-ci-cd'
            dir('simple-webapp-jenkins-ci-cd') {
              sh 'pwd && ls -la'
            }
          }
        }
      }
    }

    stage('Build') {
      steps {
        dir('simple-webapp-jenkins-ci-cd') {
          echo 'Validating folder structure and required files...'
          // Ensure scripts are executable (safety check for Windows-developed
repos)
          sh 'chmod +x validate.sh deploy.sh || true'
          // validate.sh will exit non-zero if files are missing
          sh './validate.sh'
        }
      }
    }

    stage('Test') {
      steps {
        dir('simple-webapp-jenkins-ci-cd') {
          echo 'Running simple tests (file existence) — validate.sh'
          sh './validate.sh'
        }
      }
    }

    stage('Deploy') {
      steps {
        dir('simple-webapp-jenkins-ci-cd') {
          script {
            if (env.DEPLOY_METHOD == 'docker') {
              echo "Building Docker image ${env.IMAGE_NAME}..."
              sh "docker build -t ${env.IMAGE_NAME} ."

              echo "Stopping and removing old container if exists..."
              sh "docker rm -f ${env.CONTAINER_NAME} || true"

              echo "Running new container ${env.CONTAINER_NAME} on port
${env.WEBAPP_PORT}..."
```

```
              sh "docker run -d --name ${env.CONTAINER_NAME} -p
${env.WEBAPP_PORT}:80 ${env.IMAGE_NAME}"

              echo "Deployment complete! Access the webapp at:
http://localhost:${env.WEBAPP_PORT}"
            } else {
              echo 'Deploy method: copy — running deploy.sh (may require sudo)'
              sh 'chmod +x deploy.sh || true'
              sh './deploy.sh'
            }
          }
        }
      }
    }
  }

  post {
    success {
      echo '============================================='
      echo 'Pipeline completed successfully! ✅'
      echo '============================================='
      script {
        if (env.DEPLOY_METHOD == 'docker') {
          echo "Webapp deployed and running at:
http://localhost:${env.WEBAPP_PORT}"
          echo "Container name: ${env.CONTAINER_NAME}"
          sh "docker ps | grep ${env.CONTAINER_NAME} || echo 'Container not
found'"
        }
      }
      echo '============================================='
      // Example: send email on success (requires email configured in Jenkins)
      // emailext subject: "Build Successful: ${env.JOB_NAME}
#${env.BUILD_NUMBER}", body: "Good news!"
    }
    failure {
      echo 'Pipeline failed. Inspect logs and fix the issue.'
      // Optionally send failure notification
      // emailext subject: "Build Failed: ${env.JOB_NAME}
#${env.BUILD_NUMBER}", body: "Something went wrong."
    }
  }
}
```

Dockerfile

```
# Dockerfile — serve the static site with nginx (Alpine)
# Build context: project root containing index.html, styles.css, script.js

FROM nginx:alpine

# Remove default nginx static content
RUN rm -rf /usr/share/nginx/html/*

# Copy site files into nginx html folder
```

```
COPY ./ /usr/share/nginx/html/

# Expose port 80
EXPOSE 80

# Run nginx in foreground
CMD ["nginx", "-g", "daemon off;"]
```

validate.sh

```
#!/usr/bin/env bash
# validate.sh — check required files exist before build/deploy
# Exits with non-zero status if any required file is missing

set -euo pipefail

REQUIRED=(index.html styles.css script.js)
MISSING=()

for f in "${REQUIRED[@]}"; do
  if [ ! -f "$f" ]; then
    MISSING+=("$f")
  fi
done

if [ ${#MISSING[@]} -ne 0 ]; then
  echo "Missing required files:" >&2
  for m in "${MISSING[@]}"; do echo " - $m" >&2; done
  exit 2
else
  echo "All required files present: ${REQUIRED[*]}"
fi
```

Challenges & Resolutions

- Jenkinsfile path + nested repo → set Script Path + dir()
- Script permissions on Linux → git chmod + pipeline chmod
- Docker permission denied → add Jenkins to docker group + restart
- Image pull timeout → pre-cache nginx:alpine
- Deploy method → switched from sudo copy to Docker

Lessons Learned

- Docker simplifies CI/CD deployments
- Repo structure impacts Jenkins configuration
- Windows→Linux scripts need executable flags
- Group membership changes require service restart

Conclusion

The CI/CD pipeline met all objectives, delivering 100% (4/4 stages) success with a reproducible Docker-based deployment. The project is production-ready and demonstrates a clear, maintainable Jenkins workflow for static sites.

References

- Repo: https://github.com/Punith968/simple-webapp-jenkins-ci-cd
- Jenkins Docs: https://www.jenkins.io/doc/
- Docker Docs: https://docs.docker.com/

Appendix — Key Commands

```
docker build -t simple-webapp:latest .

docker run -d --name simple-webapp-demo -p 8090:80 simple-webapp:latest

./validate.sh
```